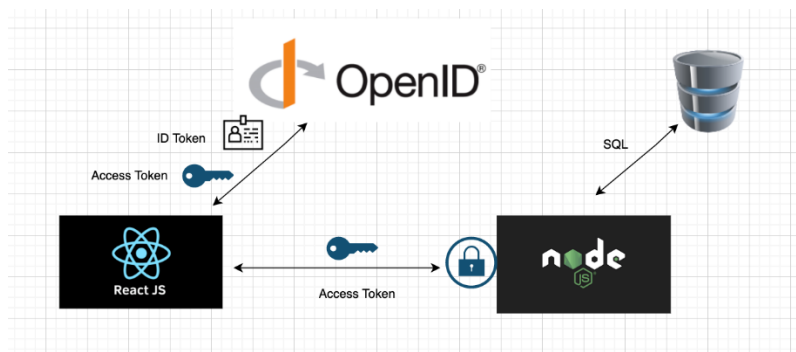


OpenID Connect and OAuth 2.0 Primer



Esta é a parte 1 de uma série de artigos que estou elaborando com o objetivo de ajudar desenvolvedores de aplicativos, engenheiros de segurança e tomadores de decisão a entender como adotar as melhores práticas do OpenID Connect e do OAuth 2.0. A citação a seguir diz tudo...

"O OAuth 2.0, nas mãos de um desenvolvedor com profundo conhecimento de segurança web, provavelmente resultará em uma implementação segura. No entanto, nas mãos da maioria dos desenvolvedores... o OAuth 2.0 provavelmente produzirá implementações inseguras." Eran Hammer, fundador original do OAuth

Background

Como observação, os termos usados neste artigo (Provedor OpenID, Provedor de Autorização e IDP) são intercambiáveis e se referem à entidade que emite e valida os tokens de segurança. OAuth 2.0 é uma estrutura focada em Autorização, originalmente projetada para permitir que aplicativos acessem APIs protegidas. Ela responde à pergunta: o que um aplicativo pode fazer? Com a implementação do OAuth, um usuário apresenta suas credenciais a um Provedor de Identidade confiável (Provedor de Autorização) que, após o consentimento do usuário, fornece um token de acesso que pode ser usado pelo aplicativo cliente para fazer chamadas de API a um servidor de recursos. O token de acesso deve ter escopo limitado, permitindo que o aplicativo cliente acesse apenas os recursos permitidos pelo proprietário do recurso.

Você pode pensar no OAuth da seguinte maneira: você vai a um hotel e mostra seu documento de identidade e o recepcionista lhe dá uma chave (token de acesso) que permite acesso a um determinado número de quarto. A chave não tem sua identidade, mas você ou qualquer outro portador dessa chave (sua esposa ou filhos) pode entrar naquele quarto usando-a, desde que a data codificada na chave ainda seja válida. Você pode pensar nessa porta como uma API em um servidor de recursos. Indo um passo além, essa chave pode ser usada para ter acesso a outros recursos no hotel, como as portas laterais, academia, piscina etc. Como observação final, suponha que você perca ou extravie a chave. Você pode ir ao recepcionista e eles invalidarão a chave antiga e emitirão uma nova porque sabem quem você é. Você apresentou credenciais no check-in para o recepcionista. Você pode pensar neles como um provedor de autorização.

A estrutura OAuth 2.0, por si só, fornece informações limitadas sobre o usuário que foi autenticado e autorizado pelo Provedor de Autorização. Ela apenas fornece autorização com um escopo que define o que o aplicativo web pode acessar no servidor de recursos em nome do usuário. Portanto, se o aplicativo precisar fornecer uma imagem ou outras informações de identidade sobre o usuário, precisará de outra maneira.

Surgiu o OpenID Connect, um padrão baseado na estrutura OAuth 2.0. O OpenID Connect fornece a identidade do usuário. O provedor de autorização fornece um token de identidade que fornece as informações autenticadas do usuário por meio de declarações, utilizando um token seguro no formato JWT, facilitando a validação e a obtenção das informações do usuário. Ele também fornece um endpoint que pode ser usado por um servidor de recursos para obter informações do perfil do usuário por meio do token de acesso. Portanto, o OpenID Connect se baseia na estrutura OAuth 2.0, fornecendo essas informações de identidade ou perfil do usuário, e não altera o funcionamento do OAuth 2.0. Essa parte permanece a mesma.

OAuth 2.0/OpenID Connect Roles

Não ignore essas terminologias, tente entender esses conceitos e isso ajudará você a entender o OpenID Connect, o OAuth e a entender diagramas complicados de arquitetura de segurança.

O documento IETF do OAuth 2.0 Authorization Framework declara os seguintes atores e suas funções nos fluxos de tokens de segurança:

Resource Server: O servidor que hospeda os recursos ou serviços protegidos que requerem autorização específica para acesso. Pode ser um servidor de API, um servidor ou serviço back-end em um data center ou até mesmo uma aplicação sem servidor, como o AWS lambda, executada em um ambiente de nuvem por trás de um gateway de API. Ao longo deste documento, o conceito de servidor de recursos descreve qualquer uma das entidades mencionadas acima. O servidor de recursos valida o token de acesso para determinar sua autenticidade e se o portador do token possui a autorização adequada para acessar o recurso.

Resource Owner/End User: quando o proprietário do recurso é uma pessoa, é o usuário final que utiliza o aplicativo cliente para acessar um recurso protegido e é capaz de conceder acesso ao aplicativo cliente. Pensando de outra forma, o aplicativo cliente obtém acesso a um recurso protegido em nome do usuário autenticado. Observação: O proprietário do recurso pode ser um usuário privilegiado. O usuário pode fazer parte de um grupo que tem acesso a esse recurso protegido específico.

Authorization/OpenID Provider: a entidade que controla as decisões de autorização. Ela valida as credenciais do proprietário do recurso e emite os tokens de segurança usados pelo aplicativo cliente para acessar o recurso protegido. Alguns o chamam de Provedor de Identidade (IDP) ou Servidor de Autorização.

Client Application/RelyingParty: O aplicativo que obtém os tokens de segurança do provedor de autorização e os utiliza para acessar um recurso ou serviço protegido. Pode ser um aplicativo legado em execução em um servidor de aplicativos, um aplicativo em contêiner em execução em um ambiente de nuvem, um aplicativo de página única (SPA) em execução em um navegador ou um aplicativo móvel em execução em um dispositivo.

Security Tokens:

Os tokens de segurança emitidos por provedores de autorização/OpenID Connect serão tokens de acesso, tokens de identidade ou tokens de atualização. Os tokens de acesso normalmente estão no formato JSON Web Token (JWT), que é independente, mas em alguns casos pode estar em um formato proprietário conhecido como tokens de referência. Já o token de ID estará sempre no formato JWT. Os tokens JWT fornecem informações ou declarações (atributos) na forma de pares nome:valor sobre o assunto (token ID) ou o que o portador do token (token de acesso) tem permissão para acessar (escopo). O formato JWT, explicado mais adiante neste artigo, permite que o aplicativo cliente e o servidor de recursos validem e analisem as informações contidas nos tokens.

What is the JSON Web Token structure?

Em sua forma compacta, os JSON Web Tokens consistem em três partes separadas por pontos (.), que são:

1. **Header** - O cabeçalho geralmente consiste em duas partes: o tipo do token, que é JWT, e o algoritmo de assinatura usado, como HMAC SHA256 ou RSA.
2. **Payload** - A segunda parte do token é a carga útil (Payload), que contém as declarações. Declarações são declarações sobre uma entidade (normalmente, o usuário) e dados adicionais. Existem três tipos de declarações: registradas, públicas e privadas.
3. **Signature** - Para criar a parte da assinatura, você precisa pegar o cabeçalho codificado, o payload codificado, um segredo, o algoritmo especificado no cabeçalho e assiná-los.

Portanto, um JWT codificado em Base64-URL tem a seguinte estrutura: x.y.z

- onde x = informações do cabeçalho conforme descrito acima
- onde y = informação de carga útil conforme descrito acima
- onde z = informação de assinatura conforme descrito acima

Um token codificado típico recebido do OP (OpenID Provider) pode ter a seguinte aparência: Observe os 3 pontos que separam o Header, Payload, e Signature

eyJhbGciOiJIUzI1NiIsInR5cGU6IjY9vaWNoYXNpdW50IiwianRpIjoibHpiMm90V1FjS0hjSE5PVGh0SXdjeCI8vbG9yYXob3N0QjkwMzEiLCJpYXQoIjE2NTExMTc1ODg2OGE5ZW0aF90aW1lIjojNXUxNzU4NTY2fQ.2S1LCot-
-2biV1emWfjgqmy4YCvURUoe1I-Er91kNE

JWT Claims

A RFC9068 define um perfil para tokens de acesso OAuth2.0 usando a estrutura JWT. Essa estrutura fornece um conjunto padrão de declarações obrigatórias e opcionais para auxiliar na interoperabilidade. As declarações (Claims) podem ser para estrutura de dados, autenticação, identidade ou autorização. As declarações (Claims) são standard/default ou podem ser personalizadas.

iss - O emissor do token. Este é o domínio do provedor de autorização que emitiu o(s) token(s). Usado pelo servidor de recursos (no caso do token de acesso) ou pelo aplicativo web cliente (no caso do token ID) para recuperar o conjunto de chaves públicas JWKS do endpoint conhecido para validar a assinatura.

aud - the audience - qualquer string que descreva o destinatário pretendido para o token. Por exemplo, pode se referir a um servidor de recursos ou serviço de back-end.

exp - tempo com data, quando o token expira.

iat - A hora em que o token de acesso foi emitido

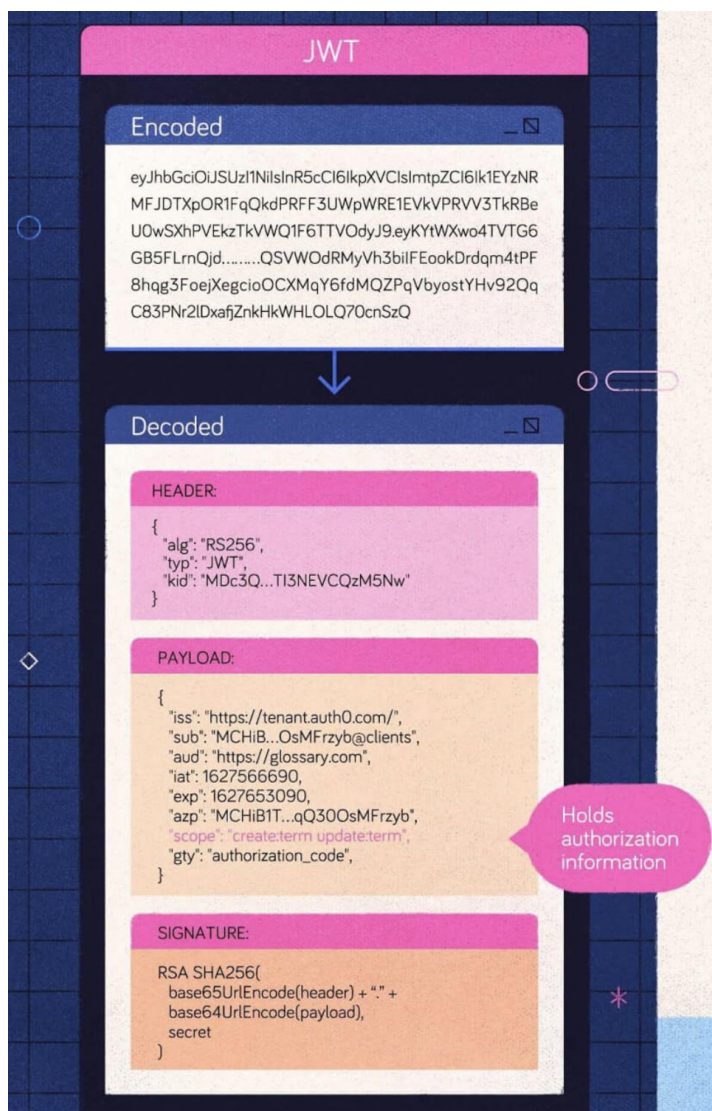
sub - subject identifier do resource owner para fluxos de authorization code. Para fluxos de client credential, o identificador deve refletir o clientID que solicita o token.

scope - O que o portador do token de acesso (geralmente o aplicativo cliente) tem permissão para fazer no servidor de recursos. Ler, Escrever, Excluir, Atualizar etc.

groups - Group geralmente mapeia para uma função. Uma reivindicação personalizada usada para determinar se o sujeito do token possui a função adequada para ser autorizado a acessar o recurso específico. Embora as funções não façam parte da estrutura OAuth 2.0, elas desempenham um papel importante no fluxo de autorização. Por exemplo: A autorização em um servidor de recursos específico consiste em funções que têm permissões sobre o que o portador do token tem permissão para fazer. Portanto, para mapeá-lo, as reivindicações de grupos mapeiam para funções e

Access Token - Tokens de acesso são credenciais usadas para acessar recursos protegidos em um servidor que os protege em algum lugar. Um token de acesso é uma string que representa uma autorização emitida ao cliente pelo provedor de autorização. Normalmente, um aplicativo cliente chamará uma API em um servidor de recursos com o token de acesso como um token portador no cabeçalho de autorização. Dependendo do servidor de autorização, a string do token de acesso pode ser analisável pelo servidor de recursos, como ao usar o Perfil JSON Web Token (JWT) para Tokens de Acesso. Eles contêm o escopo do que o aplicativo cliente tem permissão para fazer em nome do usuário e, possivelmente, declarações sobre o usuário, como grupos. Na maioria dos casos, o token de acesso será validado pelo servidor de recursos por meio de uma chamada de introspecção ao provedor de autorização para validá-lo. Ele também pode ser apresentado ao endpoint userinfo do provedor de autorização para obter a identidade do usuário (informações normalmente fornecidas pelo token id).

Sample Access Token

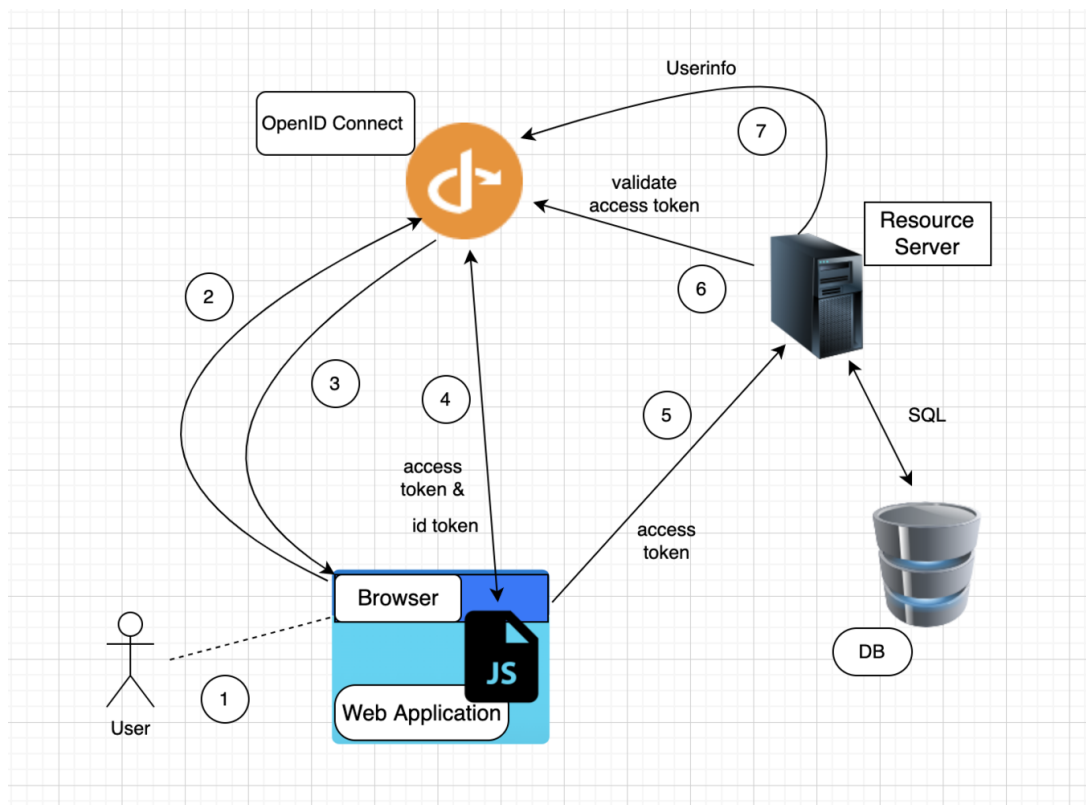


ID token - (AuthN) - Um artefato fornecido pela extensão OpenID Connect para OAuth 2.0 para permitir que usuários finais sejam autenticados. Ele fornece a identidade do usuário. O token ID é um token de segurança que contém declarações sobre a autenticação de um usuário final por um servidor de autorização. Você pode pensar no token ID como um passaporte. Você vai ao escritório do governo local e apresenta uma identidade válida, como uma carteira de motorista ou certidão de nascimento, para obter um passaporte que identifica quem você é e atribui (afirmações sobre você; peso, altura, idade, etc. Consulte o padrão OpenID Connect para obter uma lista das declarações padrão e suas descrições. Os tokens ID DEVEM ser assinados usando JWS (JSON Web Signature) e, opcionalmente, assinados e criptografados usando JWS e JWE (JSON Web Encryption) e, respectivamente, fornecendo autenticação, integridade, não-repúdio e, opcionalmente, confidencialidade.

Sample ID token

Este exemplo mostra o conteúdo de um token ID com informações sobre o usuário autenticado. Observe as declarações padrão fornecidas para o perfil do usuário em resposta ao escopo do perfil.

OpenID Connect flow with Security Tokens



1. O usuário insere uma URL na barra de endereços do navegador para o aplicativo web (cliente OAuth). O pacote JavaScript (aplicativo web) é carregado no navegador, conforme mencionado anteriormente. O aplicativo web pode apresentar uma página inicial com uma janela de login ou recursos protegidos.
2. O usuário clica em um recurso protegido e é redirecionado ao provedor de autorização para apresentar suas credenciais ou a janela de login envia as credenciais ao provedor de autorização.
3. O provedor de autorização fornecerá um código de autorização de volta ao aplicativo (cliente OAuth) por meio de um redirecionamento pelo navegador ou de uma chamada de API para o aplicativo web.
4. O cliente OAuth troca o código de autorização por um token de acesso e um token ID. O cliente OAuth pode validar o token ID e armazenar ambos os tokens no armazenamento do navegador.
5. O usuário pode então clicar em um elemento no aplicativo web que resulta em uma chamada de API para o servidor de recursos (busca) com o token de acesso como um token portador no cabeçalho de autorização.
6. O servidor de recursos validará o token de acesso. Isso pode ser feito por meio da validação da assinatura do token localmente ou por introspecção, enviando o token ao provedor de autorização para validação.
7. O token de acesso também pode ser usado para obter as informações do usuário enviando o token para o endpoint userinfo do provedor de autorização.

Getting the tokens (access and ID tokens)

Como parte do processo de autenticação, o aplicativo cliente solicita ao endpoint de autorização ou token um token de acesso, um token ID ou ambos e, em alguns casos, um refresh token, dependendo dos casos de uso e do fluxo do aplicativo. Se um refresh token for obtido na solicitação inicial, ele poderá ser usado para obter outro token de acesso após a expiração do token de acesso. Esse processo pode continuar durante toda a vida útil do refresh token.

Validating the tokens

O aplicativo cliente valida o token de ID verificando a assinatura JWT usando as chaves públicas do JWKS do provedor de autorização que emitiu o token. Além disso, declarações individuais podem ser validadas no próprio token, como emissor, data de expiração, etc.

Autenticação de token com o emissor (issuer)

O servidor para o qual o token de acesso é usado (normalmente o servidor de recursos) é responsável por validar o token de acesso. Isso pode ser feito por dois métodos diferentes:

1. Introspecção de token - Uma solicitação de postagem é enviada ao Provedor de Autorização com o token de acesso no corpo da mensagem (recomendado). A resposta retornada deve indicar a autenticidade do token.
2. local validation - Como o token JWT é autocontido e pode ter a assinatura validada para autenticidade, não é necessário usar o endpoint de introspecção. Isso pode ser vantajoso quando o tráfego de rede é uma preocupação. No entanto, vale ressaltar que a validação local não leva em conta a revogação do token pelo provedor de autorização.

Authorization of the access token

O servidor de recursos deve verificar a declaração de público (aud) no token de acesso para verificar se o público-alvo pretendido no token aponta para o servidor de recursos. O servidor de recursos também deve verificar se a declaração de grupos no token de acesso corresponde às funções necessárias para os recursos ou servidor acessados. O escopo também deve ser verificado para verificar se corresponde às permissões necessárias (por exemplo: admin, read, update, delete, put, get etc.).

Using the tokens

O token de ID é usado pelo aplicativo cliente para validar a autenticação do usuário e obter informações sobre ele. Isso pode ser útil para a personalização de um site, usando nome, e-mail, etc. Ele não deve ser encaminhado para um servidor de recursos downstream. Em vez disso, o token de acesso deve ser usado para validar se o portador do token (aplicativo cliente) tem acesso aos recursos protegidos. O token de acesso é normalmente usado como um token portador no cabeçalho de autorização ao fazer chamadas de API para um servidor de recursos local ou na nuvem, um serviço de back-end ou até mesmo uma função lambda na AWS. Refresh tokens são usados pelo aplicativo cliente para solicitar outro token de acesso caso o token inicial esteja prestes a expirar.

Além disso, tokens de acesso podem ser usados pelo servidor de recursos ou serviço de backend para obter informações do usuário sobre o usuário autenticado (se o escopo contiver openid e perfil) acessando o endpoint userinfo do provedor de autorização.

What are Scopes:

Os escopos são um aspecto fundamental para determinar o que realmente estará nos tokens de segurança mencionados acima. Basicamente, eles se dividem em dois tipos: escopos de permissão e escopos de identidade, dependendo se você está falando de OAuth 2.0 ou OpenID Connect. Se você estiver falando de OAuth 2.0, os escopos fornecem quais ações ou permissões são permitidas pelo cliente no servidor de recursos. Por exemplo: ler, gravar, excluir registros, acessar diretórios específicos, etc. A estrutura do OAuth não fornece um padrão ou detalhes específicos sobre como implementar escopos; cabe ao provedor de autorização fornecer os detalhes de implementação.

Se você estiver falando sobre o OpenID Connect, estará falando sobre escopos de identidade, ou seja, quais informações de perfil são fornecidas ao cliente no token de ID ou no endpoint userinfo. O OpenID Connect possui

escopos padrão integrados, de acordo com a especificação, e pode ter escopos personalizados opcionais que fornecem outras informações de perfil sobre o usuário: grupos, departamentos, etc. Os escopos integrados são: perfil, e-mail, endereço, telefone e OpenID (obrigatório).

What are OAuth Grants?

Uma concessão de autorização (authorization grant) é uma credencial usada pelo cliente para obter um token de acesso. A estrutura OAuth2.0 especifica os seguintes tipos de concessão no contexto de um fluxo de conexão OpenID.

authorization_code - A autorização do usuário é capturada em um código que o cliente pode trocar por um token de acesso e um token de identidade chamando o método authorization

implicit - (deprecated no OAuth 2.1) Este tipo de concessão é implícito, pois nenhum código autorizado intermediário é usado para obter o token. Quando um token de acesso é emitido por meio desse tipo de concessão, o servidor de autorização não autentica o cliente.

refresh token - usado por clientes para trocar um token de atualização por um token de acesso quando o token de acesso expira ou está prestes a expirar. Evita que o usuário precise se autenticar novamente para obter outro token de acesso.

client credentials - usadas durante service to service requests (conta de serviço) para obter um token para acessar um serviço.

Resource Owner Password - usada para obter um token de acesso fornecendo ao cliente as credenciais do usuário. O cliente envia a solicitação para obter o token do endpoint token com as credenciais do usuário (este recurso foi descontinuado pelos Padrões OAuth 2.0).

Device Flow - Uma extensão OAuth2.0 que permite que dispositivos sem recursos de navegador recuperem um token de acesso. Usado por automóveis, residências, TVs, etc.

Hybrid Flow - combinação de fluxo de código implícito e de autorização. Utiliza combinações de vários tipos de concessão, mais comumente o código id_token.

Extension Grants - fluxos de concessão de emissão de tokens não padrão, como delegação, credenciais personalizadas etc.

What are response types:

Response types - usados na solicitação de canal frontal para o endpoint do provedor de autorização. Informa o Servidor de Autorização sobre o fluxo de processamento de autorização desejado, incluindo quais tokens são retornados dos endpoints utilizados. Os seguintes tipos de resposta correspondem a cada fluxo. Observe que os fluxos são discutidos mais adiante neste artigo.

Response Types by Flow

Flow	Response Types
Authorization Code	code
Implicit	id_token
Implicit	id_token token
Hybrid	code id_token
Hybrid	code token
Hybrid	code id_token token

OpenID/OAuth2.0 connect endpoints:

Os endpoints são as URLs no provedor de autorização que o aplicativo cliente usará para enviar solicitações de autenticação do usuário e recuperar os tokens de segurança. O servidor de recursos usará os endpoints para validar um token de acesso ou recuperar o conjunto de chaves públicas que será usado para validar a assinatura do token. Os metadados ou o endpoint de discovery publicados pelo provedor podem ser usados para obter as informações exatas do URI a serem usadas para acessar o endpoint.

Authorization endpoint: - O authorization endpoint realiza a autenticação do usuário final. Isso é feito enviando o user agente ao ponto de extremidade de autorização do servidor de autorização para autenticação e autorização, usando parâmetros de solicitação definidos pelo OAuth 2.0 e parâmetros e valores de parâmetros adicionais definidos pelo OpenID Connect. Esse endpoint é comumente chamado de canal frontal, pois o user agente está envolvido na obtenção do código de autorização.

Token endpoint: - O Token endpoint é usado pela parte confiável (cliente) para obter um token de acesso, um token ID e, opcionalmente, um refresh token. O endpoint token requer autenticação da parte confiável (cliente). Esse endpoint é comumente chamado de canal secundário, pois o user agente não está envolvido na troca de tokens.

Userinfo endpoint: O Userinfo endpoint que retorna declarações sobre o usuário final autenticado. Para obter as declarações solicitadas sobre o usuário final, o cliente faz uma solicitação ao Userinfo endpoint usando um token de acesso obtido por meio da autenticação de conexão OpenID.

Introspection endpoint - A introspecção de token define um mecanismo para servidores de recursos obterem informações sobre tokens de acesso. Os servidores de recursos podem verificar a validade dos tokens de acesso e descobrir outras informações, como qual usuário e quais escopos estão associados ao token.

JWKS_URI endpoint - o endpoint para recuperar chaves públicas do JSON Web Key Store associadas às chaves privadas JWKS usadas para assinar os tokens. Elas podem ser usadas para validar o ID Token.

OIDC Discovery endpoint: O padrão OIDC configurou como opcional uma forma para os fornecedores publicarem. Este endpoint fornece uma lista de todas as URLs associadas aos endpoint para o fluxo OAuth2.0/OIDC. Desenvolvedores, equipes de segurança e automação podem usá-lo para testes e automação. <https://<url for your Authorization Server>/well-known/openid-configuration>

OpenID/OAuth2.0 - Front Channel & Back Channel

Ao discutir fluxos de conexão OpenID ou concessões OAuth, há dois tipos diferentes de conexões: front channel and back channel.

Front-channel é onde o agente do usuário (o navegador) é redirecionado para a página de login do servidor de autorização. Este também é o ponto em que a autenticação multifator (MFA) pode ser inserida no fluxo. O servidor de autorização valida as credenciais e o consentimento e, em seguida, redireciona o agente do usuário de volta para o aplicativo cliente com o código de autorização para que ele possa ser usado na troca por um token de segurança.

Back-channel é onde o aplicativo web cliente troca o código de autorização pelo token de acesso e pelo token de ID. Isso é feito em segundo plano pelo aplicativo cliente, conectando-se ao ponto final do token, não visível ao agente do usuário (navegador).

Observação: o fluxo de conexão OpenID implicit puro não vai para o back-channel token endpoint.

Client authentication with the Authorization provider

Antes de abordar os fluxos de autenticação, vale a pena observar o que acontece com a metade do canal de retorno dos fluxos. Quando uma aplicação web cliente se conecta ao provedor de autorização por meio do canal de retorno, para obter os tokens, validá-los ou até mesmo revogá-los, ela precisa ser autenticada com o provedor de autorização. Isso envolve um segredo compartilhado. Ele pode ser baseado em senha, certificado ou JWTs de chave privada. A credencial mais comum usada hoje é a senha, também conhecida como segredo. Esses segredos devem ser protegidos de alguma forma, seja por um cofre ou algum serviço que possa mantê-los.

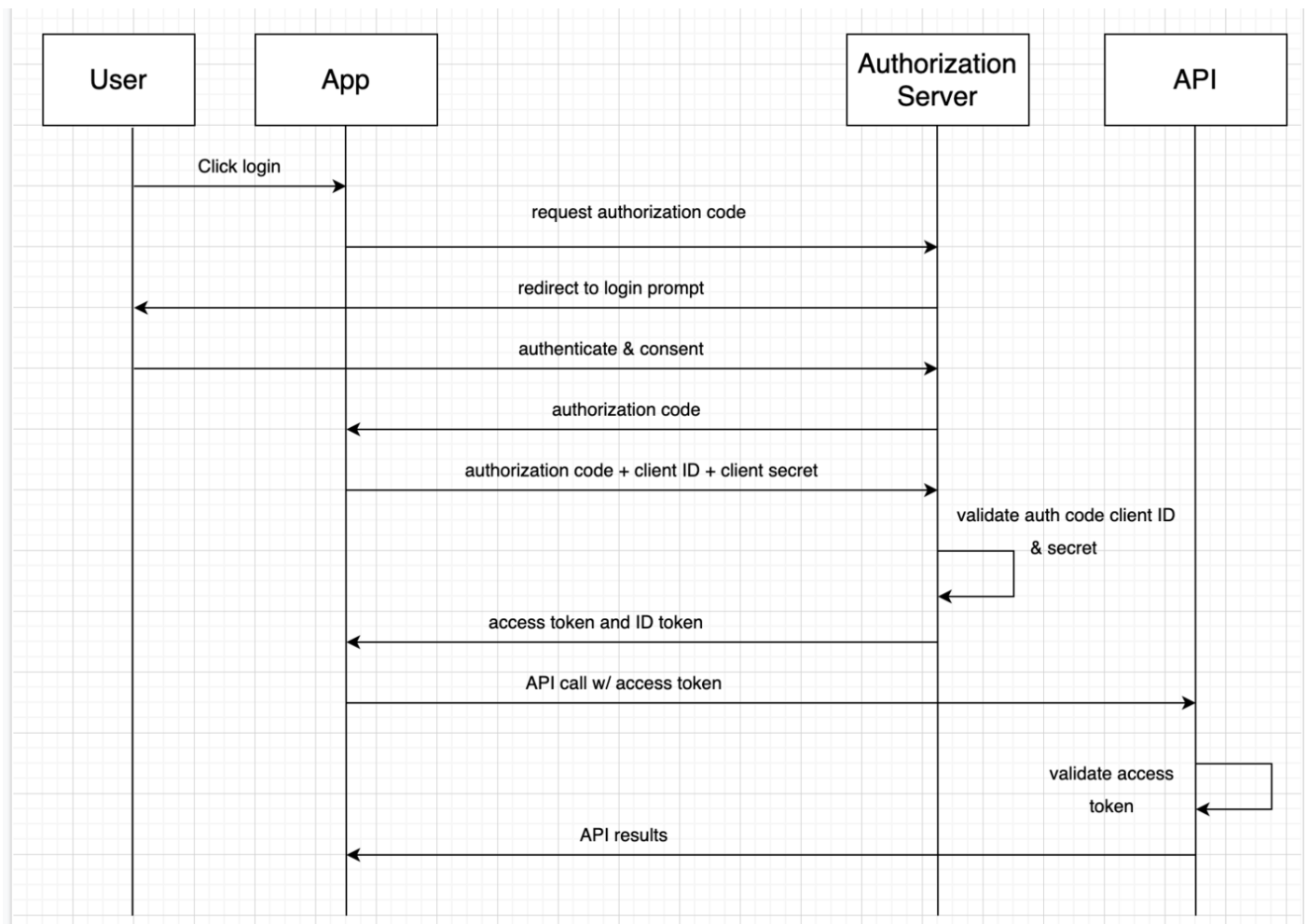
Client authorization with the Authorization Provider

Após a autenticação, o aplicativo cliente precisa ser autorizado pelo provedor de autorização para determinar qual servidor ou serviço de recurso tem permissão para usar o token de acesso. Ao solicitar um token, o aplicativo cliente deve fornecer um parâmetro de audiência que indique qual servidor ou serviço de recurso é o destinatário pretendido. O provedor de autorização analisará uma política ou alguma lista de controle de acesso para verificar se o cliente deve ter permissão para acessar o recurso e, se permitido, o token de acesso conterá a declaração de audiência com o servidor de recurso.

OpenID connect authentication flows:

Às vezes, pode ser confuso falar sobre grant types e fluxos, pois ambos parecem descrever a mesma coisa. É importante lembrar que os grant types geralmente são discutidos nas especificações do OAuth 2.0, enquanto os fluxos são mencionados no padrão OpenID Connect.

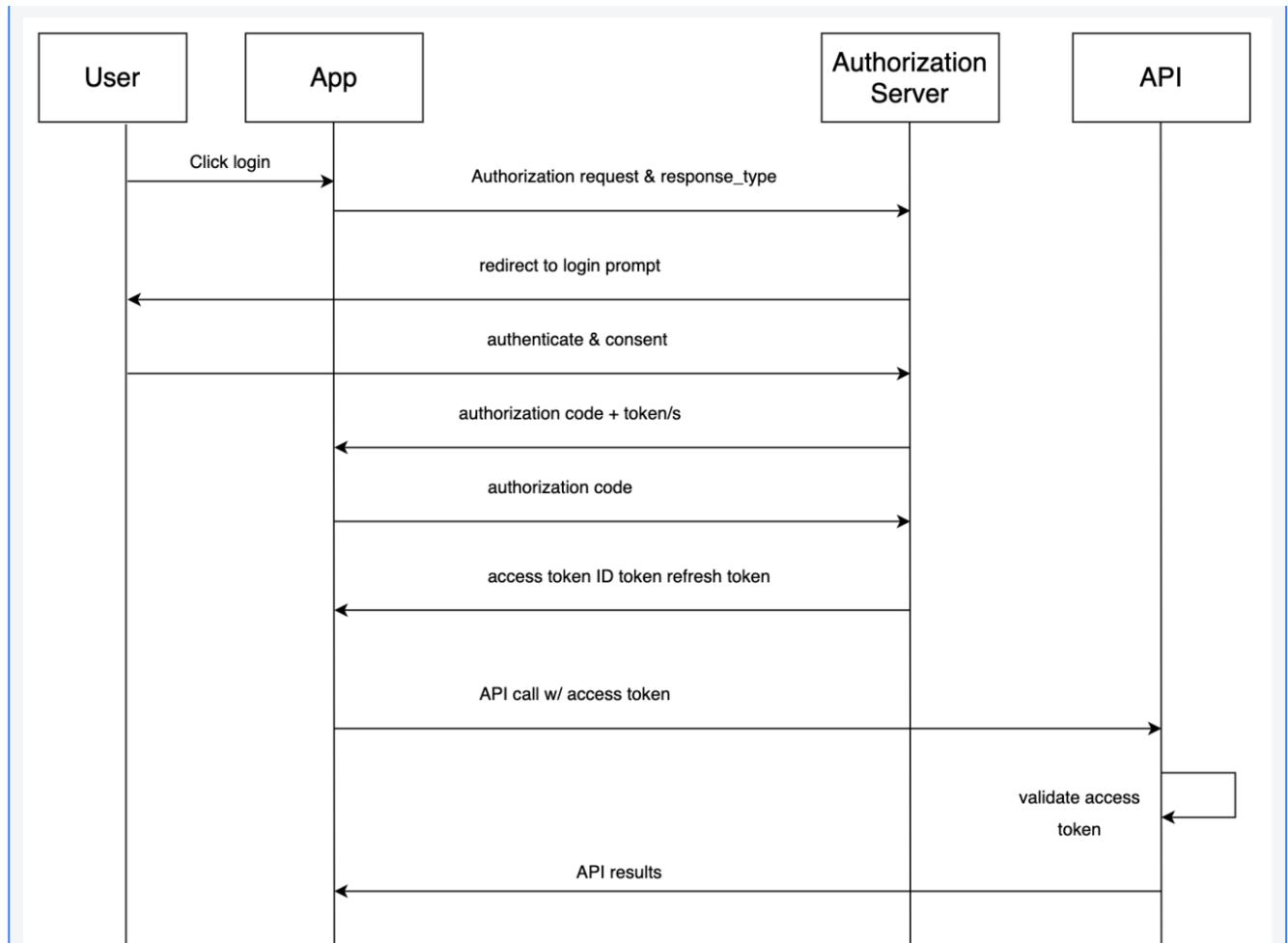
Authorization code flow (confidential clients) - Usado por aplicações web que podem armazenar um ID de cliente e um segredo com segurança. A parte confiável (aplicativo cliente) obtém um código de autorização do endpoint de autorização (front channel) do agente do usuário por meio de um redirecionamento vindo do provedor OpenID após o usuário apresentar as credenciais com sucesso. A parte confiável então troca o código de autorização por um back channel usando um segredo de cliente para um token de ID e um token de acesso e, opcionalmente, um token de atualização. O aplicativo cliente então usa o token de acesso para buscar as APIs.



Fluxo implícito - Normalmente usado por SPAs que não podem manter um segredo. A parte confiável obtém um token ID e um token de acesso diretamente do endpoint de autorização do user agent por meio de um redirecionamento vindo do provedor de autorização após o usuário apresentar as credenciais com sucesso. É estritamente um fluxo de canal frontal. O fluxo implícito não usa o back channel. O endpoint do token não é usado neste fluxo e não há tokens de atualização.

Observação: este fluxo está sendo descontinuado no OAuth 2.1 devido a preocupações de segurança relacionadas a vazamento de token de acesso e ataques de repetição. No final deste artigo, fornecerei referências para orientação sobre este fluxo. Portanto, se você estiver desenvolvendo seu SPA, considere usar o PKCE. No entanto, se seu aplicativo já o utiliza, considere eliminá-lo gradualmente nas próximas versões de software.

Hybrid Flow - O Fluxo Híbrido é um fluxo do OpenID Connect que incorpora características tanto do fluxo Implícito quanto do fluxo de Código de Autorização. Ele permite que os clientes obtenham alguns tokens diretamente do Ponto de Autorização, enquanto ainda têm a possibilidade de obter outros do Ponto de Token. Este fluxo pode ser útil se o cliente precisar realizar algum processamento com os tokens antes de trocar um código de autorização por tokens.



Trouble Shooting & Testing (Solução de problemas e testes)

Nesta seção, mostrarei como testar a configuração do seu servidor de autorização sem precisar passar pelo aplicativo cliente. Isso pode ser útil para testar a configuração do cliente OAuth antes que a equipe de desenvolvimento conclua o desenvolvimento do aplicativo cliente.

Test a basic authorization flow

Para começar, descubra seus endpoints de autorização e token acessando seu endpoint de descoberta OIDC: `https://<url for your Authorization Server>/.well-known/openid-configuration`.

Etapa 1: obtenha o código de autorização:

Para obter o código de autorização, passaremos pelo canal front-end usando a URL do navegador. Obtenha o endpoint de autorização do seu OpenID. O endpoint de descoberta é bem conhecido e mostrado nas etapas anteriores. Por exemplo, após acessar o endpoint de descoberta do OpenID, você descobrirá que o seu endpoint de autorização é:

```
https://<hostname for authorization  
server>:port>/as/authorization.oauth2/
```

insira o endpoint de autorização com os seguintes parâmetros de consulta na URL do seu navegador.

1. `client_id`: ID do cliente da sua configuração de locatário de autorização.
2. `redirect_uri`: pode ser qualquer URL para testes, conforme listado na configuração do aplicativo cliente do servidor de autorização.
3. `response_type`: use o código para o fluxo do código de autorização.
4. `scope`: o escopo openid é obrigatório. Mas você também pode especificar perfil, e-mail, etc.

É assim que você enviaria a solicitação na barra de endereço do navegador para o endpoint de autorização usando seu navegador.

```
https://<hostname of authorization  
server>:port>/as/authorization.oauth  
client_id=YOUR_CLIENT_ID&  
redirect_uri=YOUR_CALLBACK_URL&  
response_type=code&  
scope=openid?2
```

Aqui está um exemplo de solicitação para meu servidor de autorização local na barra de endereços de um navegador.

```
https://localhost:9031/as/authorization.oauth2?  
client_id=app_auth_js&response_type=code&scope=openid&redirect_u  
ri=http://localhost:3000/callback
```

O navegador deve levá-lo a um prompt de login e, assim que as credenciais forem verificadas com sucesso, ele poderá solicitar seu consentimento com base no escopo da solicitação. Geralmente, há uma configuração que pode desativar o prompt de consentimento para ignorar o consentimento exigido pelo usuário. A resposta de autorização, se bem-sucedida, retornará com um código de autorização semelhante à resposta abaixo:

```
localhost:3000/callback?  
code=gG6SQD6FPVUsUuva9ZTjwnP6tFvBfbpUR1or3G_X
```

Observação: se você não tiver uma porta escutando o número da porta para o seu URL de redirecionamento, tudo bem. Para fins de teste, você está apenas tentando obter o código de autorização.

Etapa 2: Obtenha o token do endpoint do token usando o código de autorização da etapa 1

obtenha o endpoint do token acessando seu ponto de extremidade de descoberta OIDC: `https://<url for your Authorization Server>/.well-known/openid-configuration`.

Para esta etapa, você usará o fluxo do canal de back-end para enviar uma solicitação de postagem http com o código de autorização para trocar por um token e os parâmetros necessários

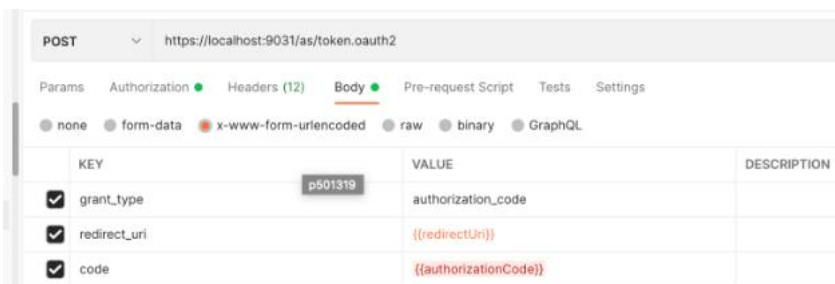
Duas ferramentas populares para acessar o back-channel são o comando curl (usado neste artigo para simplificar) e o Postman. Há inúmeras coleções do Postman na internet que possuem solicitações configuradas e prontas para uso para acessar um servidor de autorização. Você só precisa atualizar as variáveis de ambiente para incluir o ambiente específico do seu servidor de autorização.

usando algo como o comando curl ou postman mostrado abaixo.

1. url - endpoint do token (use o endpoint `./well-known/openid-configuration` para obter o endpoint do token.
2. header: content-type: application/x-www-form-urlencoded
3. grant_type: sempre use `authorization_code` ao usar o código de autorização para obter os tokens
4. client_id: este será o mesmo `client_id` usado na etapa anterior para obter o código de autorização.
(Observação: seu provedor de autorização pode autenticar com base no `client id` e no segredo. Portanto, você pode usar `clientId` e segredo no corpo ou como uma autenticação básica dentro do cabeçalho de autorização.
5. code: seu código de autorização obtido na etapa anterior nº 1
6. redirect_uri: o mesmo `redirect_uri` usado para obter o código de autorização da etapa nº 1.

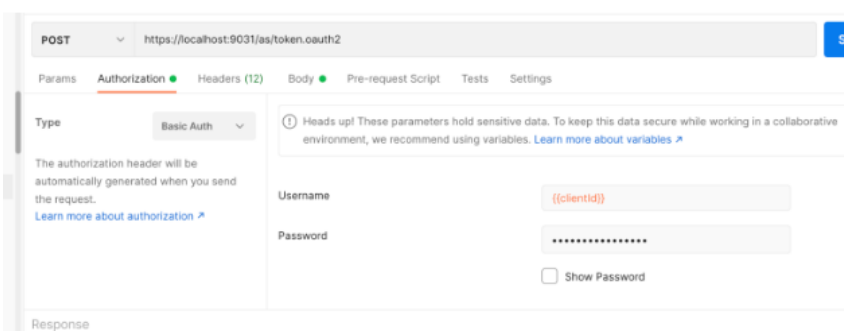
```
curl -k --request POST \
  --url 'https://localhost:9031/as/token.oauth2' \
  --header 'content-type: application/x-www-form-urlencoded' \
  --data grant_type=authorization_code \
  --data client_id=app_auth_js \
  --data code=gG6SQD6FPVUsUuva9ZTjwnP6tFvBfbpUR1or3G_X \
  --data 'redirect_uri=http://localhost:3000/callback'
```

Carteiro: clique na aba Corpo e insira o URI de redirecionamento usado na solicitação do canal frontal e o código de autorização retornado pelo provedor de autorização na solicitação do canal frontal.



KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> grant_type	authorization_code	
<input checked="" type="checkbox"/> redirect_uri	{{redirectUri}}	
<input checked="" type="checkbox"/> code	{{authorizationCode}}	

Clique na aba Autorização para inserir os parâmetros básicos de autenticação: ID do cliente e segredo para autenticação com o provedor de autorização. Observe que você também pode usar o cliente e o segredo no Corpo.



Type: Basic Auth

Username: {{clientId}}

Password:

☐ Show Password

Se a solicitação de token não apresentar erros, você deverá ver a resposta com os tokens. Nesse cenário, o servidor de autorização retorna um `access_token` e um `token ID`.

access token:

eyJhbGciOiJSUzI1NiIsImtpZCI6ImxsIiwicGkuYXRRTjoiM2l6dSJ9.eyJzY29wZSI6WyJvcGVuaWQiXSwiY2xpZW50X2lkIjoieWNfb2ljX2NsaWVudCIsImFnaWQ.IoiJUXUP6VDlxvVhHYndtQ2x10EHGRIRuanhYNWFibVlyZYisILVzZXJuYWllIjoIdXNlcidWIiwiz3jdXBziJoibXlpZHAIcLCPcmdoYWllIjoIdXNlcidWIiwizXhwIjoxNjYyMTcnInMyfQ.Ad88i3_ZjnOpBg3qRyw2okAto6TuT0QzHPNbArve52jX.QmgkfHJKLObhW6C3A4TLT-kJT5uNLd6L6XUVJg9_yk_9615MMQLXquU15-udFx0q-QAlAJ3GP5VLdwGrHDafPTKsbYb2VBHzdT_6u_nFb8o7QY1glumS-2ongZZLJ6PGZgeM3-Jgjr-kkJPXbrklblfyWGd8LVocwcpX41KKX3KEEKw0l3farceawOIevvtB0ptdpTj.VWTkd1Rycvro8pukXXzc2TYOjOsMPmPKst6KHhkoeY-3Bo2rnl_QuOpwn0Div_nClvhe7bfvc0c20njxhmFiSRlwyn5yg

corte e cole os tokens de acesso e ID no link jwt.io para obter os tokens decodificados

dados de carga útil do decodificador jwt.io

HEADER: ALGORITHM & TOKEN TYPE	<pre> { "alg": "RS256", "kid": "k1", "pi.atm": "3izu" } </pre>
PAYLOAD: DATA	<pre> { "scope": ["openid"], "client_id": "ac_oic_client", "agid": "WQzzT9qYXGbwmc1e8HFFtjnxX5aHmYrg", "Username": "user.0", "groups": "myidp", "OrgName": "user.0", "exp": 1662175732 } </pre>
VERIFY SIGNATURE	<pre> RSASHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), Public Key in SPK) </pre>

id token:

[illegible]

dados de payload do decodificador jwt.io

HEADER: ALGORITHM & TOKEN TYPE
<pre>{ "alg": "HS256" }</pre>
PAYLOAD: DATA
<pre>{ "sub": "user.0", "aud": "ac_oic_client", "jti": "2eAv0IYznexJNh5x5i72cP", "iss": "https://localhost:9031", "iat": 1662168529, "exp": 1662168829, "auth_time": 1662168508 }</pre>
VERIFY SIGNATURE
<pre>HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret) <input type="checkbox"/> secret base64 encoded</pre>

Test the Implicit Flow

Como não há conexão de canal de retorno com o fluxo implícito, essas etapas exigirão apenas o fluxo de mensagens do canal frontal. Há apenas uma etapa necessária para obter os tokens de acesso e/ou ID. Esta é a maneira mais rápida de obter tokens se a configuração do cliente no provedor de autorização permitir a concessão implícita.

As etapas são semelhantes às da etapa 1 do fluxo de código de autorização básico, com exceção do `response_type` e do valor `nonce`. Nesse caso, o `response_type` seria o `token id_token` para obter ambos os tokens de acesso de volta. Para o valor `nonce`, você pode inserir uma string aleatória.

Descubra seus endpoints de autorização e token acessando seu endpoint de descoberta OIDC: `https://<url do seu Servidor de Autorização>/.well-known/openid-configuration`.

```
https://<hostname for authorization
server>:port>/as/authorization.oauth2/
```

insira o ponto de extremidade de autorização com os seguintes parâmetros de consulta na URL do seu navegador.

`client_id`: ID do cliente da sua configuração de locatário de autorização.

`redirect_uri`: pode ser qualquer URL para testes, conforme listado na configuração do aplicativo cliente do servidor de autorização.

`response_type`: use o código para o fluxo do código de autorização.

`scope`: o escopo `openid` é obrigatório. Mas você também pode especificar perfil, e-mail, etc.

`nonce`: valor de string usado para associar uma sessão de cliente a um token de ID e para mitigar ataques de repetição.

Aqui está um exemplo de uma solicitação de autorização usando fluxo implícito, usando o tipo de resposta = `token id_token` para solicitar o token de acesso e o token de identidade para meu servidor de autorização local na barra de endereços de um navegador.

Observação: também o valor nonce como um parâmetro de solicitação é necessário. (pode ser qualquer string para teste)

```
https://localhost:9031/as/authorization.oauth2?  
client_id=app_auth_js&response_type=token id_token  
&scope=openid&redirect_uri=http://localhost:3000/callback&nonce=  
12345
```

Se a solicitação de autorização não apresentar erros, você deverá ver a resposta com os tokens. Nesse cenário, o servidor de autorização retorna um access_token, id_token. Observação: não há tokens de atualização com o throw implícito.

```
http://localhost:3000/callback#access_token=os911DZ7IoDGnkT4BEbf  
TBT8UiSk&id_token=eyJhbGciOiJSUzI1NiIsImtpZCI6Ij0TVVA3Q2poamRRRE  
VSUTRkeTdkUkVbVNFQ5J9.eyJzdWIiOiJ1c2VyLjAiLCJhdwQiOiJhcHBfYXV0a  
F9qcyIsImp0aSI6Im4xVm92WE1Md3V1c1RsaUJUTFFTS2wiLCJpc3MiOiJodHRwc  
zovL2xvY2FsaG9zdDo5MDMxIiwiaWF0IjoxNjUzOTE3ODY4LCJleHAiOiJlE2NTM5M  
TgxNjgsImF1dGhfdGltZSI6MTY1MzkxNzg2Niwiibm9uY2UiOiIxMjM0NTYiLCJhd  
F9oYXNoIjoiaDFZSTdIYk5paEZY0FpBOUd1RmoydyJ9.KYm8mIMSN-  
HcVe6KtmE7ZuCrOI16VNbT8j77-  
TLLafH92IXUYJLrNSeL4CCKihzsHCaH0ucCF_yiq08GE8v7S-xogELyyGDBf65W-  
5oPTFvYyiqxj85ECKYkPKZQ94o8QhKqKE6Uaog5kkH7qx1xPJSVNAuvGC0GqG7s1  
T5iCo7HhTvqqCB1z4liFbkhwYJdY_fo8iasBRhsNrZ2G30oK9iv7FCE1eyQ5LoSi  
AAB975LcLR8LZJEyq1Pd1_j90yPjeNq7MBzssPulQ1IJvqPjMAIqvxyHHHVAKghe  
aP9QCYHRsLcfeEyPIk3TtjL9r05a7FKeLs6zlnHpoidwtqm_w&token_type=Bea  
rer&expires_in=59
```

Se você cortar e colar esta resposta para

Esperamos que estes últimos exemplos ajudem a demonstrar como testar a configuração do seu aplicativo cliente OIDC Connect com um Provedor de Autorização.

As referências a seguir são recursos valiosos para aprimorar seu conhecimento e seguir as melhores práticas de segurança usando o protocolo e os padrões OpenID Connect Auth2.0.